

# Informe Laboratorio 2

## Sección 1

Tobías Guerrero Chequepán  
e-mail: tobias.guerrero\_c@mail\_udp.cl

Septiembre de 2023

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades</b>	<b>3</b>
2.1. Levantamiento de docker para correr DVWA (dvwa) . . . . .	3
2.2. Redirección de puertos en docker (dvwa) . . . . .	4
2.3. Obtención de consulta a replicar (burp) . . . . .	4
2.4. Identificación de campos a modificar (burp) . . . . .	6
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	7
2.6. Obtención de al menos 2 pares (burp) . . . . .	9
2.7. Obtención de código de inspect element (curl) . . . . .	12
2.8. Utilización de curl por terminal (curl) . . . . .	12
2.9. Demuestra 4 diferencias (curl) . . . . .	14
2.10. Instalación y versión a utilizar (hydra) . . . . .	15
2.11. Explicación de comando a utilizar (hydra) . . . . .	15
2.12. Obtención de al menos 2 pares (hydra) . . . . .	17
2.13. Explicación paquete curl (tráfico) . . . . .	17
2.14. Explicación paquete burp (tráfico) . . . . .	18
2.15. Explicación paquete hydra (tráfico) . . . . .	19
2.16. Mención de las diferencias (tráfico) . . . . .	21
2.17. Detección de SW (tráfico) . . . . .	21

## 1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA

(Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de inspect elements de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en vulnerabilities/brute. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?

## 2. Desarrollo de actividades

Antes de iniciar el levantamiento de docker para correr DVWA, se instala docker con su repositorio y respectivos paquetes, mediante los siguientes comandos ingresados por terminal:

```

1 sudo apt update
2 sudo apt upgrade
3 sudo apt-get install curl apt-transport-https ca-certificates software-
    properties-common
4 curl -fsSL https://download.docker.com/linux/ubuntu/gpg
5 sudo apt-key add -
6 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/
    linux/ubuntu $(lsb_release -cs) stable"
7 sudo apt update
8 apt-cache policy docker-ce
9 sudo apt install docker-ce
10 sudo systemctl status docker

```

Listing 1: Comandos para la instalación de docker

### 2.1. Levantamiento de docker para correr DVWA (dvwa)

Una vez que ya se encuentra instalado docker, se procede a buscar la imagen de DVWA y se descarga, esto se realiza mediante los siguientes comandos:

```

1 sudo docker search web-dvwa
2 sudo docker pull vulnerables/web-dvwa

```

Listing 2: Instalación imagen dvwa

Una vez descargada la imagen se monta el contenedor con los siguiente parámetros:

```
1 sudo docker run --rm -it -p 8080:80 vulnerables/web-dvwa
```

Listing 3: Levantando contenedor con imagen dvwa

A continuación se detalla en la terminal este proceso:

```

tobias@tobias-NBLK-WAX9X:~$ sudo docker run --rm -it -p 8080:80 vulnerables/web-dvwa
[sudo] contrasena para tobias:
[+] Starting mysq...
[+] Starting mariadb database server: mysq... ly to suppress this message
[ ok ] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
. . OK
=> /var/log/apache2/access.log <== AH00163: Apache/2.4.25 (Debian) configured -- resumi
=> /var/log/apache2/error.log <== AH00094: Command line: '/usr/sbin/apache2'
[Tue Sep 12 20:13:49.341016 2023] [mpm_prefork:notice] [pid 302] AH00163: Apache/2.4.25 (Debian) configured -- resumi
ng normal operations
[Tue Sep 12 20:13:49.341016 2023] [core:notice] [pid 302] AH00094: Command line: '/usr/sbin/apache2'
=> /var/log/apache2/other_vhosts_access.log <== .username=smithy&password=arthur&Login=Login
=> /var/log/apache2/access.log <== 172.17.0.1 - - [12/Sep/2023:20:13:52 +0000] "GET /vulnerabilities/brute/?username=smithy&password=arthur&Login=Login
HTTP/1.1" 302 343 "http://localhost:8080/vulnerabilities/brute/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36"
172.17.0.1 - - [12/Sep/2023:20:14:01 +0000] "GET /vulnerabilities/brute/?username=admin&password=cream&Login=Login
HTTP/1.1" 302 343 "http://localhost:8080/vulnerabilities/brute/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36"
172.17.0.1 - - [12/Sep/2023:20:14:19 +0000] "GET /vulnerabilities/brute/?username=pablo&password=cream&Login=Login
HTTP/1.1" 302 343 "http://localhost:8080/vulnerabilities/brute/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36"
172.17.0.1 - - [12/Sep/2023:20:14:46 +0000] "GET /vulnerabilities/brute/?username=l337&password=cream&Login=Login
HTTP/1.1" 302 343 "http://localhost:8080/vulnerabilities/brute/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36"
172.17.0.1 - - [12/Sep/2023:20:15:02 +0000] "GET /vulnerabilities/brute/?username=smithy&password=cream&Login=Login
HTTP/1.1" 302 343 "http://localhost:8080/vulnerabilities/brute/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36"

```

Figura 1: Levantando docker con dvwa

## 2.2. Redirección de puertos en docker (dvwa)

De la figura 1 se logra apreciar que se inicia el contenedor, el cual recibe como parámetros:

- Sudo: Se utiliza para ejecutar comandos de Docker con privilegios de superusuario. Esto es necesario en muchos sistemas para interactuar con Docker.
- docker run: Esto indica que se crea y ejecuta un nuevo contenedor Docker a partir de una imagen.
- -rm: Esta opción indica que el contenedor debe eliminarse automáticamente después de que se detenga. Esto es útil para limpiar automáticamente los contenedores temporales después de su uso.
- -it: Estas dos opciones se combinan para habilitar una terminal interactiva dentro del contenedor. -i significa “interactivo” y -t asigna un pseudo TTY (terminal) para interactuar con el contenedor.
- -p 8080: Esta opción establece un mapeo de puertos que permite que el puerto 80 dentro del contenedor (donde se está ejecutando un servicio web) sea accesible desde el puerto 8080 en el sistema anfitrión.
- vulnerable/web-dvwa: Esto especifica la imagen de Docker que se utilizará para crear el contenedor. En este caso, se utiliza la imagen “vulnerable/web-dvwa”, que esta relacionada con Damn Vulnerable Web Application (DVWA), una aplicación deliberadamente vulnerable utilizada para practicar pruebas de seguridad web.

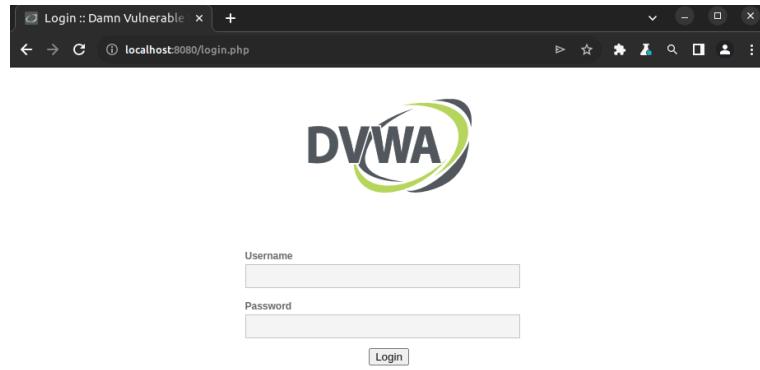


Figura 2: DVWA corriendo en contenedor

## 2.3. Obtención de consulta a replicar (burp)

Para utilizar Burpsuite primero se descarga el archivo desde la página portswigger, para luego instalar el archivo mediante el siguiente comando:

### 2.3 Obtención de consulta a replicar (burp)

## 2 DESARROLLO DE ACTIVIDADES

```
1 sudo sh burpsuite_community_linux_v2023_9_4.sh
```

Listing 4: Instalando Burpsuite

A continuación se muestra la herramienta abierta y lista para interceptar:

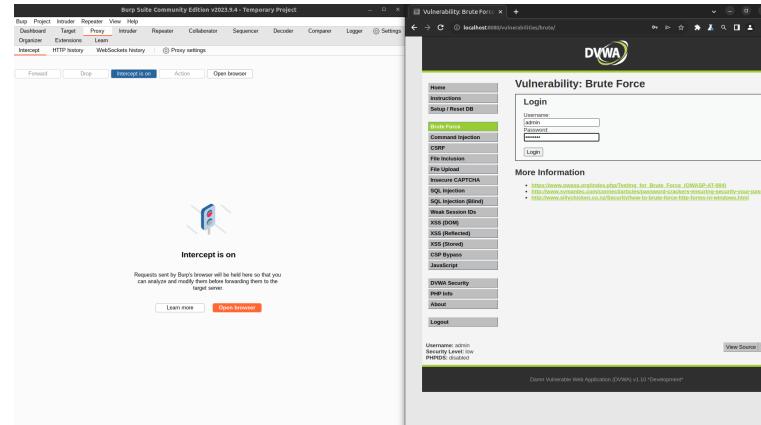


Figura 3: Burp antes de interceptar

Una vez todo listo, se inicia la intersección en DVWA (en este se ingresan las credenciales de admin y password) para que las solicitudes enviadas por el navegador de Burp se guarden aquí, para poder obtener la consulta a replicar y posteriormente a modificarla. A continuación se muestra la consulta replicada:

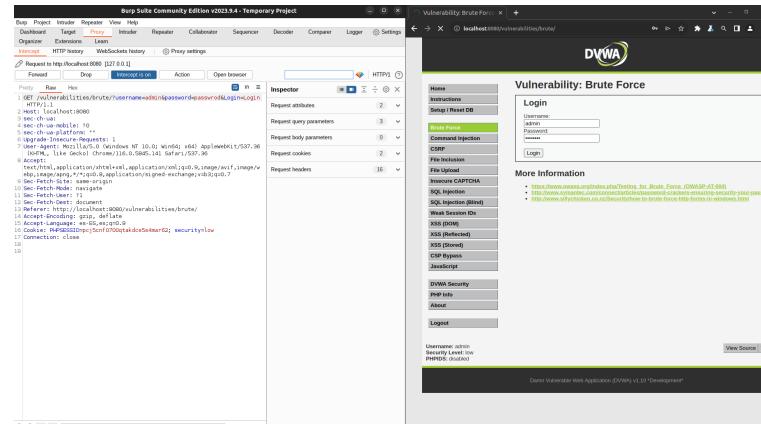


Figura 4: Consulta a replicar

## 2.4 Identificación de campos a modificar (burp) 2 DESARROLLO DE ACTIVIDADES

### 2.4. Identificación de campos a modificar (burp)

Una vez obtenida la consulta, con click derecho enviaremos esta consulta a el campo de “**Intruder**”, luego en intruder se selecciona el tipo de ataque en cluster bomb y posteriormente seleccionaremos los campos a modificar para realizar el ataque, que en este caso serán “**admin**” y “**password**”. A continuación se logra apreciar la selección:

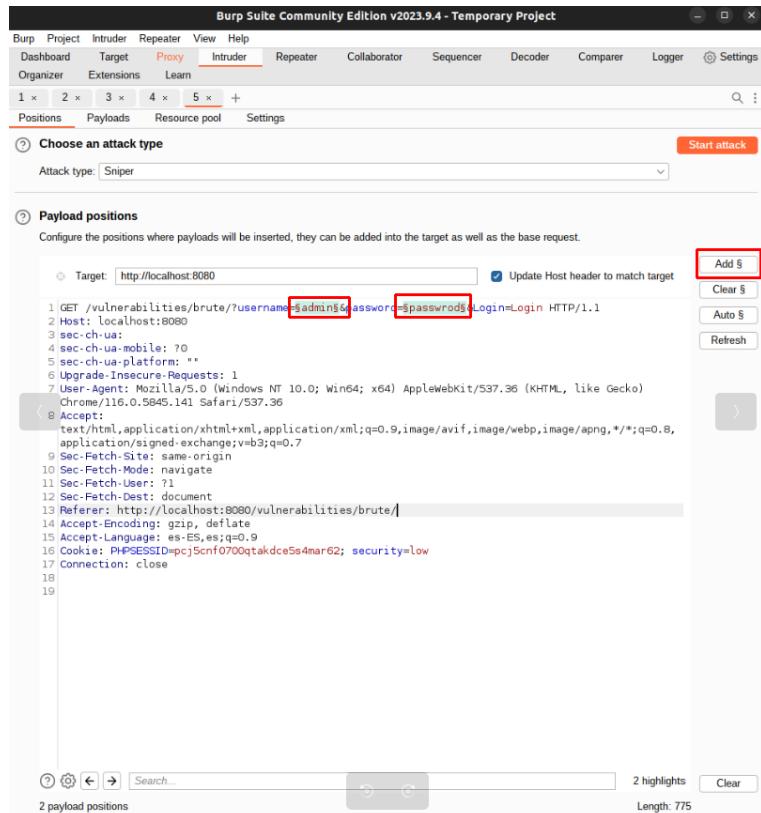


Figura 5: Campos que serán modificados

De aquí se logra apreciar que el campo de **admin** queda asociado al **payload 1** y la **password** al **payload 2**. Para posteriormente a cada payload cargarle su respectivo diccionario.

## 2.5 Obtención de diccionarios para el ataque (burp) DESARROLLO DE ACTIVIDADES

### 2.5. Obtención de diccionarios para el ataque (burp)

A continuación se muestran los diccionarios utilizados y como se le cargaron a la herramienta en sus respectivos payloads.

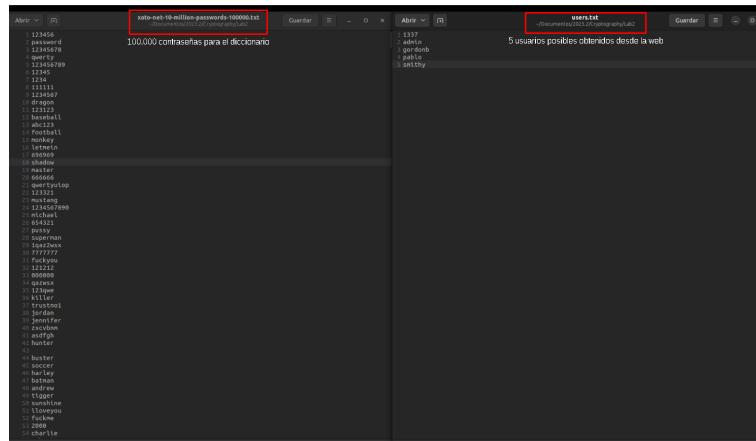


Figura 6: Diccionarios a utilizar para el ataque

De aquí es importante mencionar que la web resulta bastante vulnerable, ya que como se conocen las credenciales de admin y password e iniciar sesión, la página al retornar una imagen e inspeccionar esta, se puede acceder a la ruta “<http://localhost:8080/hackable/users/>”, de donde se trajeron los usuarios posibles.

The screenshot shows a web browser displaying the Apache server index for the '/hackable/users' directory. The address bar shows the URL '127.0.0.1:8080/hackable/users/'. The page title is 'Index of /hackable/users'. Below the title is a table with columns 'Name', 'Last modified', and 'Size Description'. The table lists five files:

Name	Last modified	Size Description
<a href="#">Parent Directory</a>	-	-
<a href="#">1337.jpg</a>	2018-10-12 17:44	3.6K
<a href="#">admin.jpg</a>	2018-10-12 17:44	3.5K
<a href="#">gordonb.jpg</a>	2018-10-12 17:44	3.0K
<a href="#">pablo.jpg</a>	2018-10-12 17:44	2.9K
<a href="#">smithy.jpg</a>	2018-10-12 17:44	4.3K

At the bottom of the page, the text 'Apache/2.4.25 (Debian) Server at 127.0.0.1 Port 8080' is visible.

Figura 7: Usuarios posibles

Además de esto, el diccionario de las passwords se obtuvo a través de un repositorio de passwords de Github, a continuación se puede cliquear en: “Diccionario de passwords”

## 2.5 Obtención de diccionarios para el ataque (burp) DESARROLLO DE ACTIVIDADES

Una vez obtenidos los diccionarios correspondientes, estos fueron cargados en sus respectivos payloads con los archivos .txt:

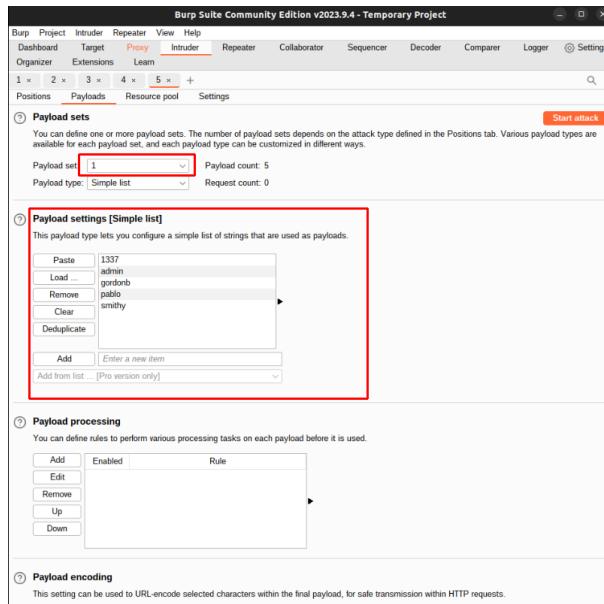


Figura 8: Payload 1 con users.txt (usuarios)

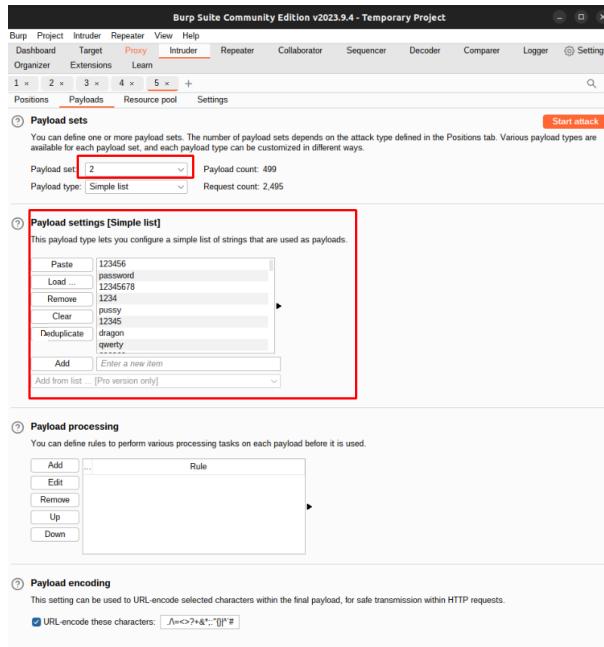


Figura 9: Payload 2 con xato-net-10-million-passwords-100000.txt (contraseñas)

## 2.6. Obtención de al menos 2 pares (burp)

Una vez completados los campos, se inicio el ataque, así obteniendo los siguientes pares de usuario/contraseña válidos:

The screenshot shows the Burp Suite interface with a list of captured requests and a browser window displaying a login page.

**Burp Suite List View:**

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
0	1337	123466	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
1	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
2	gordonb	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
3	pablo	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
4	smitty	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
5	admin	123466	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
6	gordonb	123466	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
7	pablo	123466	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
8	smitty	123466	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
9	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
10	smitty	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
11	1337	1234678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
12	admin	1234678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
13	gordonb	1234678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
14	pablo	1234678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
15	smitty	1234678	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
16	1337	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
17	admin	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	

**Burp Suite Render View:**

The browser window shows a login page titled "Vulnerability: Brute Force". It has fields for "Username" and "Password", and a "Login" button. A message below says "Welcome to the password protected area admin".

Figura 10: Primer par obtenido

Del primer par obtenido se tiene:

- Usuario: admin
- Password: password

The screenshot shows the Burp Suite interface with a list of captured requests and a browser window displaying a login page.

**Burp Suite List View:**

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
42	admin	696969	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
43	gordonb	696969	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
44	pablo	696969	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
45	smitty	696969	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
46	1337	mustang	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
47	admin	mustang	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
48	gordonb	mustang	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
49	pablo	mustang	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
50	smitty	mustang	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
51	1337	lemmie	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
52	admin	lemmie	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
53	gordonb	lemmie	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
54	pablo	lemmie	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
55	smitty	lemmie	200	<input type="checkbox"/>	<input type="checkbox"/>	4702	
56	1337	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
57	admin	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
58	gordonb	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
59	pablo	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	

**Burp Suite Render View:**

The browser window shows a login page titled "Brute Force". It has fields for "Username" and "Password", and a "Login" button. A message below says "Welcome to the password protected area pablo".

Figura 11: Segundo par obtenido

Del segundo par obtenido se tiene:

- Usuario: pablo
- Password: letmein

## 2.6 Obtención de al menos 2 pares (burp)

## 2 DESARROLLO DE ACTIVIDADES

The screenshot shows the Burp Suite interface. At the top, there's a menu bar with 'Attack', 'Save', 'Columns', 'Results', 'Positions', 'Payloads', 'Resource pool', and 'Settings'. Below the menu is a table titled 'Filter: Showing all items' with columns: Request, Payload 1, Payload 2, Status code, Error, Timeout, Length, and Comment. The table lists 100 rows of captured requests. Row 82 is highlighted with a red box, showing 'gordonb' in the 'Payload 1' column and 'abc123' in the 'Payload 2' column. Below the table, there are tabs for 'Request', 'Response', and 'Render'. The 'Render' tab is selected, displaying a 'Login' form with fields for 'Username' and 'Password', and a 'Login' button. To the left of the render area is a sidebar with various attack types: 'Setup / Reset DB', 'Brute Force' (which is selected), 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection', 'SQL Injection (Blind)', 'Weak Session IDs', and 'XSS (DOM)'.

Figura 12: Tercer par obtenido

Del tercer par obtenido se tiene:

- Usuario: gordonb
- Password: abc123

This screenshot is similar to Figure 12, showing the Burp Suite interface with the 'Brute Force' attack type selected. The table at the top shows 17 rows of captured requests. Row 8 is highlighted with a red box, showing 'smithy' in the 'Payload 1' column and 'password' in the 'Payload 2' column. The 'Render' tab is selected, displaying a 'Login' form with fields for 'Username' and 'Password', and a 'Login' button. The sidebar on the left includes 'Brute Force' (selected), 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection', 'SQL Injection (Blind)', 'Weak Session IDs', and 'XSS (DOM)'.

Figura 13: Cuarto par obtenido

Del cuarto par obtenido se tiene:

- Usuario: smithy
- Password: password

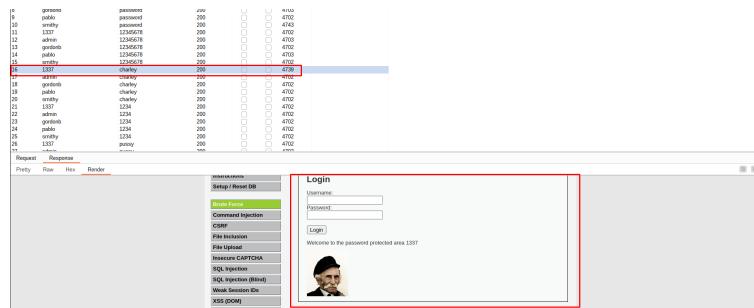


Figura 14: Quinto par obtenido

Del quinto par obtenido se tiene:

- Usuario: 1337
- Password: charley

Es importante mencionar que como utilice 100.000 contraseñas no deje que el ataque finalizará al 100 % debido a que eran muchas combinaciones posibles. Lo finalicé a penas me di cuenta de que ya tenía todas las contraseñas de los usuarios que conocía. Por otra parte se podía identificar que las combinaciones eran correctas gracias al largo, ya que cuando las credenciales son correctas la página responde una imagen, esto hacía que las correctas tuvieran un mayor largo versus las incorrectas. A continuación se adjunta un ejemplo de cuando era incorrecta la combinación:

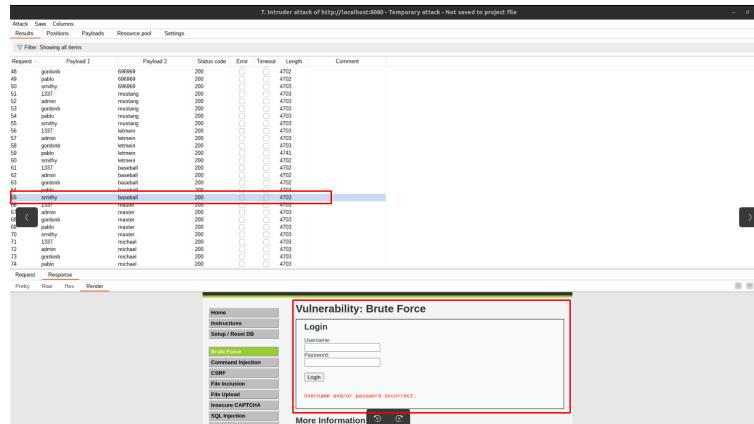


Figura 15: Combinación incorrecta

## 2.7. Obtención de código de inspect element (curl)

Para obtener el código de la consulta, inspeccionaremos la página, en el apartado de red y luego nos loguearemos en el recuadro de fuerza bruta, pero probando dos casos; el primero será con credenciales correctas y el segundo con credenciales incorrectas, con el fin de que la terminal retorne la solicitud HTTP para cada caso, la cual podrá ser copiada en un comando cURL que replica la solicitud HTTP, para posteriormente utilizarlo en la terminal. A continuación se aprecia este proceso:

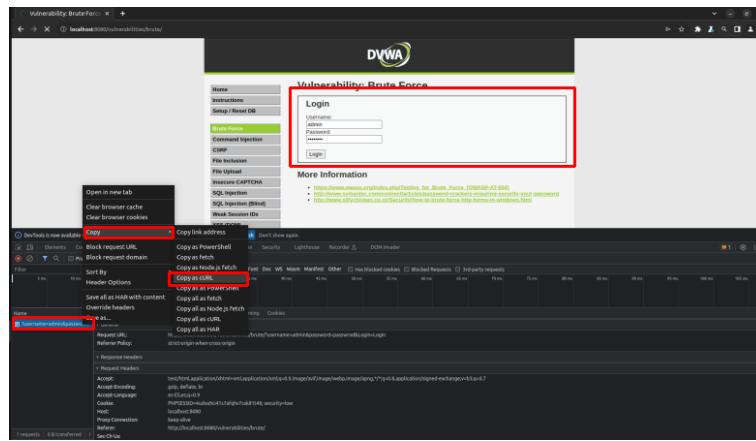


Figura 16: Obtención del código de la consulta

## 2.8. Utilización de curl por terminal (curl)

Una vez obtenido los códigos estos fueron pegados en la terminal de linux. A continuación se detalla este proceso:

- Consulta con combinaciones correctas

```
tobias@tobias-OptiPlex-5090: ~ curl http://localhost:8088/vulnerabilities/brute/?username=admin&password=password&Login=Login \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Cookie: PHPSESSID=bgpm6scfufvlqv14r62z80; security_low=' \
-H 'Referer: http://localhost:8088/vulnerabilities/brute/?username=slethy&password=password&Login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36' \
-H 'sec-ch-ua: "Not A Brand";v="1"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Ubuntu"' \
--compressed
```

Figura 17: Combinaciones correctas

- Consulta con combinaciones incorrectas

```
tobias@tobias-NBLK-WAX9... ~ tobias@tobias-NBLK-WAX9... ~ tobias@tobias-NBLK-WAX9... ~ tobias@tobias-NBLK-WAX9... ~ tobias@tobias-NBLK-WAX9...
tobias@tobias-NBLK-WAX9:~$ curl 'http://localhost:8080/vulnerabilities/brute/?username=admin&password=ola&Login=Login#'
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 1333
Date: Mon, 10 Jun 2019 17:17:20 GMT
Connection: keep-alive
Set-Cookie: PHPSESSID=7csz10l7s13gl0t0l3fsmud6; security='low' 
```

Extracto de la terminal mostrando una respuesta HTML de un sitio web vulnerable a un ataque de fuerza bruta. La URL es http://localhost:8080/vulnerabilities/brute/?username=admin&password=ola&Login=Login#. La respuesta muestra un formulario de login con los campos 'Username' y 'Password' prellenados con 'admin' y 'ola' respectivamente, y el botón 'Login' marcado como 'submit'.

Figura 18: Combinaciones incorrectas

En ambos casos se obtuvieron respuestas en html, un ejemplo de extracto de cada caso a continuación:

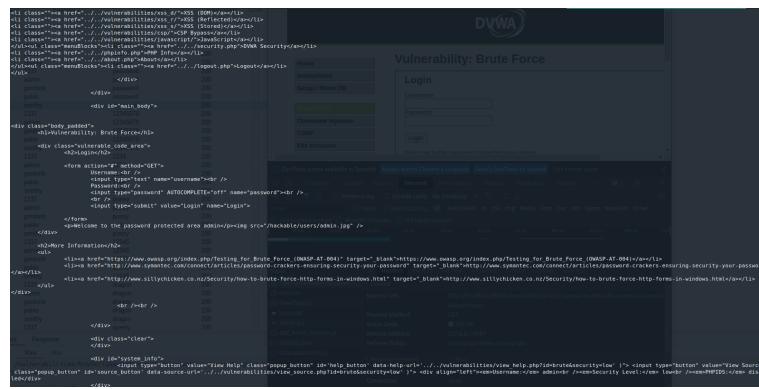


Figura 19: Respuestas combinaciones correctas

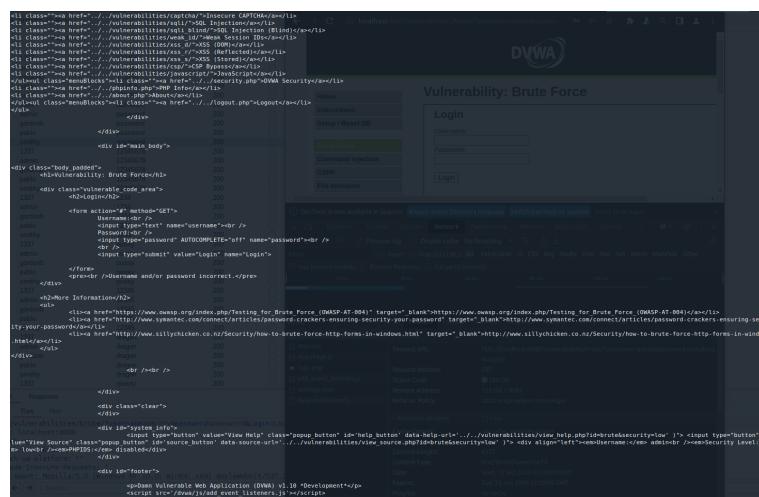


Figura 20: Respuesta combinaciones incorrectas

## 2.9. Demuestra 4 diferencias (curl)

Una vez ingresados ambos comandos se obtuvieron diferentes respuestas en la terminal, dentro de las cuales se pueden rescatar estas principales 4 diferencias:

\*A la izquierda tendremos el cURL correcto y a la derecha el cURL incorrecto

```
tobias@tobias-NBLK-WAXX:~/Documentos/2023_2/Cryptography/Labs$ curl -s http://localhost:8080/vulnerabilities/brute?username=admin&password=password&Login=Login# \ host:8080/vulnerabilities/brute?username=mala&password=mala&Login=Login# \
```

Figura 21: Primera diferencia

La primera diferencia que se logra apreciar es que en ambos casos el request url será diferente, ya que tendrán diferentes parámetros ingresados.



Figura 22: Segunda diferencia

La segunda diferencia es el mensaje de respuesta, ya que como se logra apreciar, cuando el cURL es correcto se retorna “Welcome to the password protected area admin” y cuando el cURL es incorrecto se retorna “Username and/or password incorrect.”.

```
tobias@tobias-NBLK-WAXX:~/Documentos/2023_2/Cryptography/Labs$ curl -s http://localhost:8080/vulnerabilities/brute?username=admin&password=password&Login=Login# \ host:8080/vulnerabilities/brute?username=mala&password=mala&Login=Login# \
```

Figura 23: Tercera diferencia

La tercera diferencia es que el campo de Referer es distinto, este campo se establece en la URL de la página de inicio de sesión anterior, desde la cual se originó la solicitud actual, es decir, para el cURL correcto antes de ingresar admin y password se ingreso admin y aaa, para el cURL incorrecto antes de ingresar admin y mala, se ingreso admin y password.

```
:0 $ PHPSESSID=3bsos965k0t2edqkf/in0s+9dp2; security=low" \ s'http://localhost:8080/vulnerabilities/brute?username=admin&password=password&Login=Login# \
```

Figura 24: Cuarta diferencia

La cuarta diferencia es con respecto al content-length, cuando se trata de un cURL correcto este es más largo, pero cuando se trata de un cURL incorrecto este será más corto.

## 2.10. Instalación y versión a utilizar (hydra)

Para poder instalar hydra se utilizaron los siguientes comandos:

```

1 sudo apt-get update
2 sudo apt install -y python php curl wget git nano
3 cd $HOME
4 sudo git clone https://github.com/vanhauser-thc/thc-hydra
5 cd $HOME/thc-hydra
6 sudo ./configure
7 sudo make
8 sudo make install

```

Listing 5: Comandos para la instalación de hydra

De aquí se obtiene que la versión instalada para hydra es la 9.6, a continuación se aprecia esto:

A terminal window titled 'tobias...' showing the command 'hydra -version'. The output is: 'tobias@tobias-NBLK-WAX9X:~\$ hydra -version' followed by 'hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these \*\*\* ignore laws and ethics anyway)'.

Figura 25: Versión de hydra

## 2.11. Explicación de comando a utilizar (hydra)

Una vez instalado hydra, se procede a utilizar el siguiente comando para poder realizar el ataque:

```

1 hydra -L users.txt -P xato-net-10-million-passwords-100000.txt "http-get-
  form://127.0.0.1:8080/vulnerabilities/brute/:username=^USER^&password=^
  PASS^&Login=Login:H=Cookie\: PHPSESSID=7cs2i017s13gl0tc0lt3fsmud6;
  security=low;:F=Username and/or password incorrect." -I

```

Listing 6: Comandos para la instalación de hydra

El comando consta de lo siguiente:

- **hydra:** Este es el comando principal de Hydra.
- **-L users.txt:** Esto especifica el archivo “users.txt” como la lista de nombres de usuario que Hydra utilizará en el ataque. Hydra probará cada nombre de usuario en esta lista.
- **-P xato-net-10-million-passwords-100000.txt:** Esto especifica el archivo “xato-net-10-million-passwords-100000.txt” como la lista de contraseñas que Hydra utilizará en el ataque. Hydra probará cada contraseña en esta lista.
- **http-get-form://:** Especifica el protocolo de solicitud HTTP que se utilizará.
- **127.0.0.1:8080/vulnerabilities/brute/:** Es la URL objetivo a la que Hydra enviará las solicitudes HTTP. En este caso, apuntar a una página de inicio de sesión en el puerto 8080 de la dirección IP 127.0.0.1 (localhost).
- **:username=^USER^&password=^PASS^&Login=Login:** Esta parte de la URL define cómo se enviarán los nombres de usuario y contraseñas en la solicitud HTTP. Los lugares marcados con “^USER^” y “^PASS^” serán reemplazados por los nombres de usuario y contraseñas de las listas especificadas anteriormente.
- **:H=Cookie\ PHPSESSID=7cs2i0l7s13gl0tc0lt3fsmud6; security=low;:** Define las cookies que se deben incluir en la solicitud HTTP. En este caso, se incluyen las cookies “PHPSESSID” y “security” con sus valores correspondientes. Estos valores fueron obtenido nuevamente por inspección de la página en el apartado de red.
- **:F=Username and/or password incorrect.:** Esta parte define una cadena que indica si la autenticación ha fallado. En este caso, Hydra buscará esta cadena en la respuesta HTTP para determinar si las credenciales son válidas o no.
- **-I:** Esta opción le dice a Hydra que realice un ataque de fuerza bruta de manera interactiva, lo que significa que mostrará el progreso y los resultados en tiempo real en la terminal.

## 2.12. Obtención de al menos 2 pares (hydra)

Una vez corrido el comando por la terminal, esta retorno lo siguiente:

```
tobias@tobias-NBLK-WAX9X:~/Documentos/2023.2/Cryptography/Lab2$ hydra -L users.txt .P xato-net-1
0-million-passwords-100000.txt "http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username^
USER=&password^&Login=Login:H=Cookie: PHPSESSID=7cs2i0l7s13gl0tc0lt3fsmud6; security=low;
:F=Username and/or password incorrect." -I
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak . Please do not use in military or sec
ret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws a
nd ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-09-13 12:34:08
[INFORMATION] escape sequence \: detected in module option, no parameter verification is perform
ed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 500000 login tries (l:5/p:100000), -31250 tr
ies per task
[DATA] attacking http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username^USER=&password^
^PASS^&Login=Login:H=Cookie: PHPSESSID=7cs2i0l7s13gl0tc0lt3fsmud6; security=low;:F=Username and
/or password incorrect.
[8080][http-get-form] host: 127.0.0.1 login: 1337 password: charley
[8080][http-get-form] host: 127.0.0.1 login: admin password: password
[8080][http-get-form] host: 127.0.0.1 login: gordonb password: abc123
[8080][http-get-form] host: 127.0.0.1 login: pablo password: letmein
[8080][http-get-form] host: 127.0.0.1 login: smithy password: password
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-13 12:35:02
```

Figura 26: Obtención de pares de combinaciones

Las cuales corresponde a las mismas obtenidas de Burp, es decir:

- user: admin / password: password
- user: pablo / password: letmein
- user: gordonb / password: abc123
- user: smithy / password: password
- user: 1337 / password: charley

## 2.13. Explicación paquete curl (tráfico)

Para hacer la captura de curl, se inició primero la captura de wireshark y luego se introdujo el código ya mencionado en el ítem *Obtención de código de inspect element (curl)* en la terminal, una vez realizada la captura enviando los códigos de combinación correcta y combinación incorrecta, se hizo un filtro por http para ver que pasaba con los paquetes, obteniendo lo siguiente:

## 2.14 Explicación paquete burp (tráfico)

## 2 DESARROLLO DE ACTIVIDADES

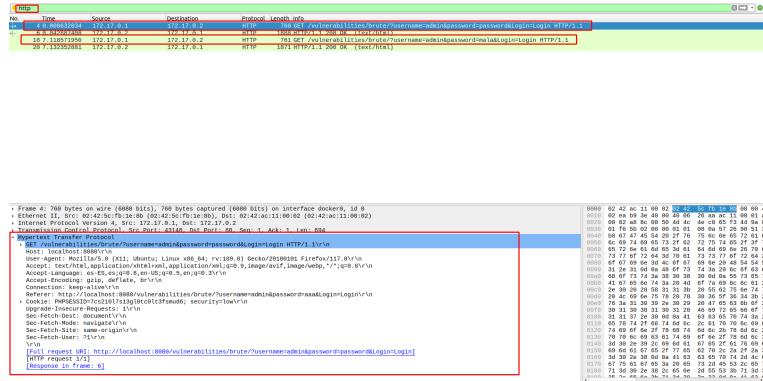


Figura 27: Paquetes curl capturados con filtro http

En cuanto a los paquetes se logra apreciar lo siguiente:

- En cada paquete va especificado el request url, en donde se indica el usuario y la contraseña. Solo tenemos dos, ya que cURL solo hace un intento, por ende uno es de las credenciales correctas y el otro de las incorrectas.
- En el campo Hypertext Transfer Protocol tenemos la cookie utilizada, el host, el user-agent y el refer del cual ya se habló anteriormente.

En cuanto al trafico general de cURL se obtuvo lo siguiente:

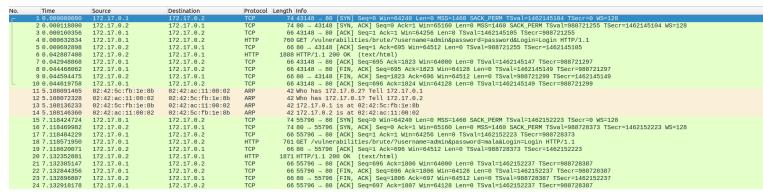


Figura 28: Trafico curl capturado

De aquí podemos apreciar que el tráfico se encuentra organizado, con un alto volumen de paquetes TCP que se guían en base a Three-Way Handshake, seguido de respuestas http coherentes. Por otra parte se logra apreciar que curl realiza solo un intento al realizar el ataque.

## 2.14. Explicación paquete burp (tráfico)

Para hacer la captura de burp, se inició primero la captura de wireshark y luego se inició el ataque con los diccionarios ya cargados, luego se hizo un filtro por http para ver que pasaba con los paquetes, obteniendo lo siguiente:

## 2.15 Explicación paquete hydra (tráfico)

## 2 DESARROLLO DE ACTIVIDADES

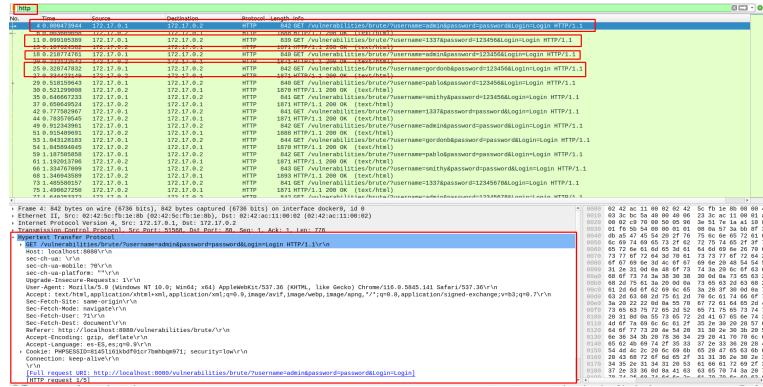


Figura 29: Paquetes burp capturados con filtro http

En cuanto a los paquetes se logra apreciar lo siguiente:

- En cada paquete va especificado el request url, en donde se indica el usuario y la contraseña, por ende se logran apreciar múltiples paquetes que van con estos campos diferentes, debido a que burp esta probando las combinaciones de cada diccionario.
- En el campo Hypertext Transfer Protocol tenemos la cookie utilizada, el host, el user-agent y el refer del cual ya se habló anteriormente.

En cuanto al trafico general de burp se obtuvo lo siguiente:

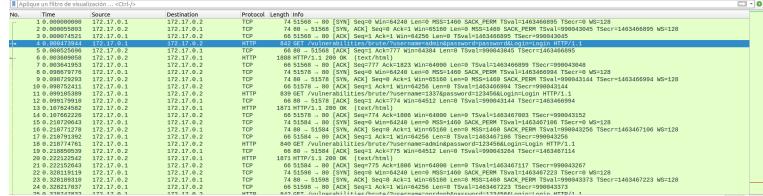


Figura 30: Trafico burp capturado

De aquí podemos apreciar que el trafico es similar a un cURL solo que en esta ocasión aparecen muchos más paquete de http debido a los múltiples intentos de combinaciones. Es importante mencionar que burp si utiliza un diccionario muy grande es bastante lento realizando el ataque.

## 2.15. Explicación paquete hydra (tráfico)

Para hacer la captura de hydra, se inició primero la captura de wireshark y luego se inició el ataque con el comando ya mencionado, luego se hizo un filtro por http para ver que pasaba con los paquetes, obteniendo lo siguiente:

## 2.15 Explicación paquete hydra (tráfico)

## 2 DESARROLLO DE ACTIVIDADES

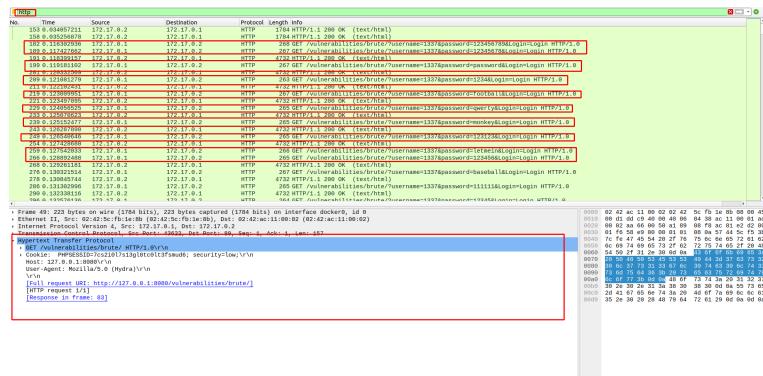


Figura 31: Paquetes hydra capturados con filtro http

En cuanto a los paquetes se logra apreciar lo siguiente:

- En cada paquete va especificado el request url, en donde se indica el usuario y la contraseña, eso si habían paquetes en donde estos parámetros no iban, más abajo se muestra una foto de esto.
- En el campo Hypertext Transfer Protocol tenemos la cookie utilizada, el host y el user-agent.

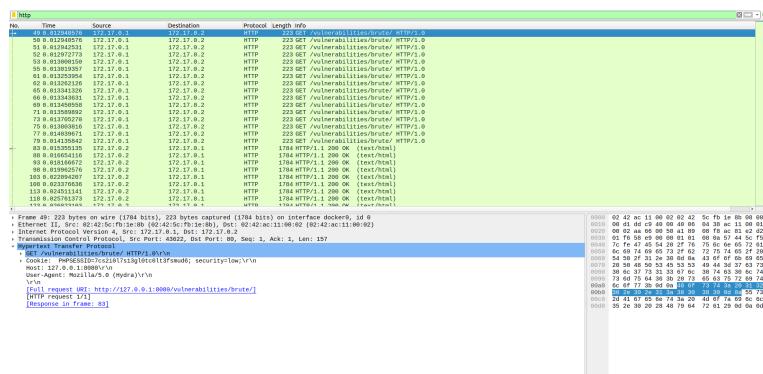


Figura 32: Paquetes sin user ni password

En cuanto al trafico general de hydra se obtuvo lo siguiente:

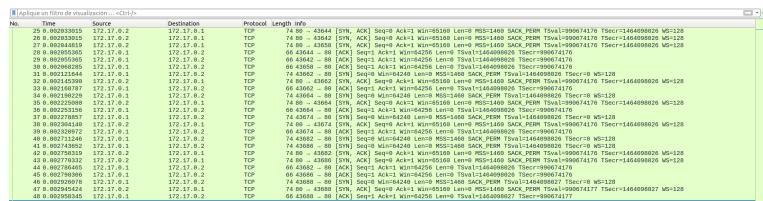


Figura 33: Trafico hydra capturado

De aquí podemos apreciar que el trafico tiene un alto volumen de paquetes TCP e hydra es muy rápido realizando el ataque, a pesar de usar .txt grandes.

## 2.16. Mención de las diferencias (tráfico)

Las diferencias entre los tráficos fueran las siguientes:

- Hydra es mucho más rápido al generar el trafico y las combinaciones versus burp, esto se debe a que Hydra está diseñado específicamente para ataques de fuerza bruta y utiliza técnicas de paralelismo para probar múltiples combinaciones en poco tiempo.
- Hydra genera un alto trafico de TCP versus cURL y burp.
- Hydra y burp hacen multiples intentos pero cURL hace solo uno.
- Hydra y cURL se utilizan principalmente para enviar solicitudes HTTP personalizadas y automatizadas, Burp Suite es una suite de herramientas de seguridad más amplia que incluye un proxy interceptador. Burp Suite permite un control más detallado sobre las solicitudes y respuestas, lo que facilita el análisis y la manipulación de las interacciones entre el cliente y el servidor. Por lo tanto, Burp Suite tiende a generar un tráfico más organizado y permite una mayor interacción durante las pruebas.
- Burp destaca por su capacidad para interceptar y manipular el tráfico en tiempo real. Puede pausar solicitudes, modificar parámetros y luego enviar solicitudes modificadas al servidor.

## 2.17. Detección de SW (tráfico)

La manera en que se pudo detectar la herramienta que se uso para realizar el ataque, fue mediante el **user-agent**, ya que este es un encabezado que se encuentra en los paquetes de solicitud HTTP enviados por un cliente (como un navegador web o una aplicación) a un servidor web durante una comunicación a través del protocolo HTTP. Este encabezado proporciona información sobre el cliente que está realizando la solicitud, especialmente el nombre y la versión del software del cliente. Es por esto que a continuación se detalla como se identificó cada uno:

## 2.17 Detección de SW (tráfico)

## 2 DESARROLLO DE ACTIVIDADES

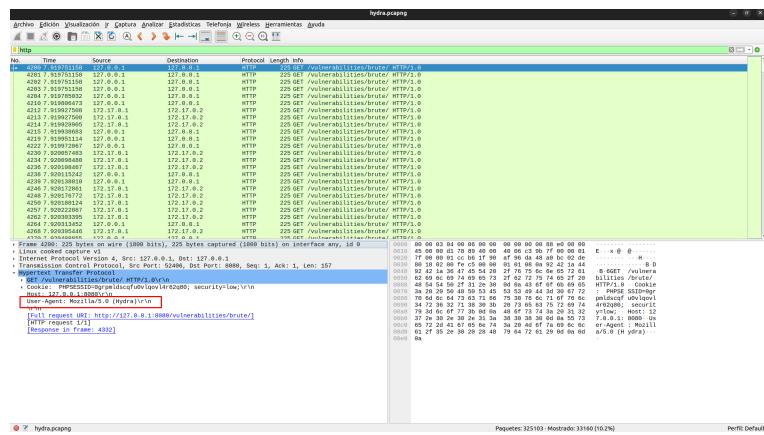


Figura 34: SW para hydra

De la figura anterior podemos identificar que en el campo user-agent dice **Mozilla/5.0 (Hydra)**, esto nos indica que el cliente que realiza la solicitud es hydra.

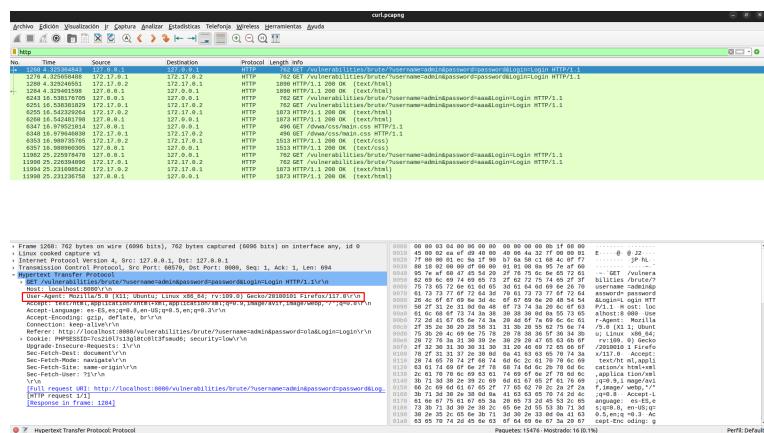


Figura 35: SW para curl

De la figura anterior podemos identificar que en el campo user-agent dice **Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/117.0**, esto nos indica que la solicitud http se originó desde un cliente que está utilizando un navegador web Firefox en un sistema Ubuntu Linux de 64 bits, lo cual representa a la herramienta cURL, ya que desde la terminal linux se corrió el comando.

## 2.17 Detección de SW (tráfico)

## 2 DESARROLLO DE ACTIVIDADES

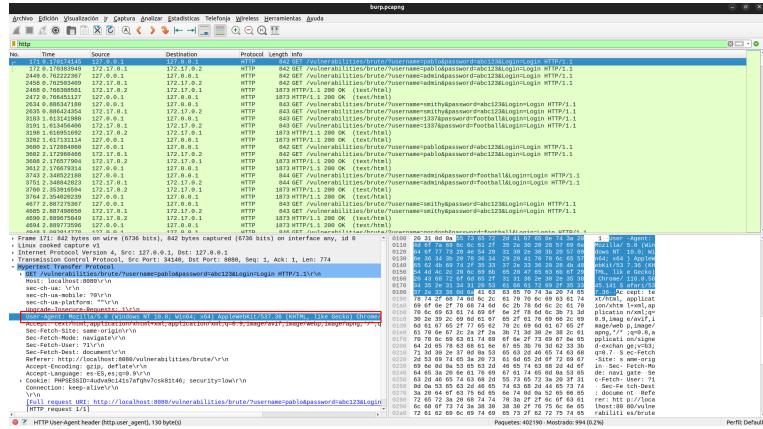


Figura 36: SW para burp

De la figura anterior podemos identificar que en el campo user-agent dice **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36**, esto nos indica que la solicitud http se originó desde un cliente que está utilizando el navegador web Google Chrome en un sistema Windows 10 de 64 bits, lo cual representa a burp, ya que este para realizar la intersección abre el navegador chrome y mediante este realiza el ataque.

## Conclusiones y comentarios

A lo largo del laboratorio se llevaron a cabo diversas actividades que proporcionaron una visión sobre la importancia de la seguridad en aplicaciones web y las herramientas disponibles para evaluarla. A través del uso de Docker, se destacó la utilidad de contener aplicaciones en entornos controlados, lo que facilita el desarrollo y las pruebas sin impactar el sistema host. Este enfoque es esencial en el mundo de la seguridad cibernética, ya que garantiza que las pruebas no afecten inadvertidamente a otros sistemas.

El ataque de fuerza bruta realizado con Burp Suite puso de relieve la eficacia de esta herramienta en la identificación de vulnerabilidades relacionadas con contraseñas débiles. Sin embargo, también se evidenció que la velocidad de un ataque de fuerza bruta está limitada por factores como la red y el rendimiento del servidor. Esto subraya la necesidad de optimizar y proteger las contraseñas de las aplicaciones.

El uso de cURL para realizar accesos válidos e inválidos al formulario ilustró cómo esta herramienta puede emplearse para enviar solicitudes HTTP personalizadas directamente desde la línea de comandos. Sin embargo, se observó que cURL se limitó a un solo intento en su solicitud, lo que indica que no se implementó un mecanismo de fuerza bruta en la solicitud cURL.

Por otro lado, el ataque de fuerza bruta con Hydra demostró ser una herramienta eficiente y rápida para encontrar contraseñas débiles o comunes. La capacidad de Hydra para realizar múltiples intentos por segundo lo convierte en una opción poderosa para realizar pruebas de seguridad en aplicaciones web.

Finalmente, las actividades del laboratorio subrayaron la relevancia de realizar pruebas de seguridad exhaustivas en aplicaciones web y resaltaron la importancia de elegir las herramientas adecuadas para abordar los desafíos específicos. La seguridad de una aplicación es fundamental y debe ser abordada con medidas sólidas, como la implementación de contraseñas seguras y la protección contra ataques de fuerza bruta.